REVISED VERSION

PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 4 : H04Q 3/00

A1

(11) International Publication Number: WO 89/ 04578

(43) International Publication Date: 18 May 1989 (18.05.89)

(21) International Application Number: PCT/US88/02390

(22) International Filing Date: 14 July 1988 (14.07.88)

(31) Priority Application Number: 119,330

(32) Priority Date: 10 November 1987 (10.11.87)

(33) Priority Country: US

(71) Applicant: ECHELON SYSTEMS [US/US]; 727 University Avenue, Los Gatos, CA 95030 (US).

(72) Inventors: MARKKULA, Armas, C., Jr. ; 250 Family Farm Road, Woodside, CA 94062 (US). SANDER, Wendell, B. ; 112 Harwood Court, Los Gatos, CA 95032 (US). EVAN, Shabtai ; 13648 Westover Drive, Saratoga, CA 95070 (US). SMITH, Stephen, B. ; 611 Navarrq Drive, Scotts Valley, CA 95066 (US). TWITTY, William, B. ; 141 Lance Court, Santa Cruz, CA 95065 (US).

(74) Agents: TAYLOR, Edwin, H. et al.; Blakely, Sokoloff, Taylor & Zafman, 12400 Wilshire Boulevard, Seventh Floor, Los Angeles, CA 90025 (US).

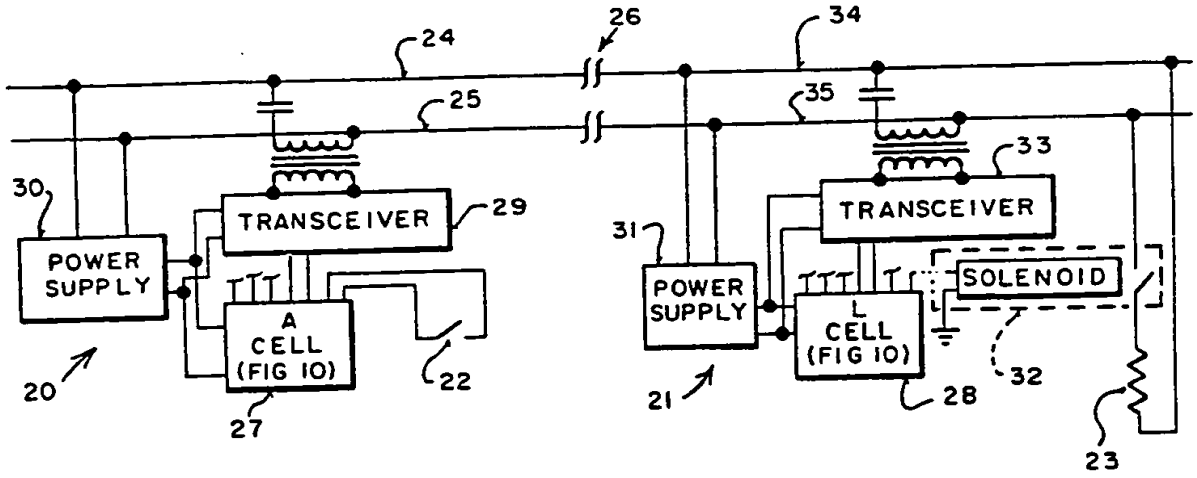
(81) Designated States: AT, AT (European patent), AU, BB, BE (European patent), BG, BR, CH, CH (European patent), DE, DE (European patent), DK, FI, FR (European patent), GB, GB (European patent), HU, IT (European patent), JP, KP, KR, LK, LU, LU (European patent), MC, MG, MW, NL, NL (European patent), NO, RO, SD, SE, SE (European patent), SU.

Published

With a revised version of the international search report. Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

Date of publication of the revised version of the international search report: 15 June 1989 (15.06.89)

(54) Title: NETWORK AND INTELLIGENT CELL FOR PROVIDING SENSING, BIDIRECTIONAL COMMUNICATIONS AND CONTROL



(57) Abstract

A network for providing sensing, communications and control. In the present invention a plurality of intelligent cells (27, 28), each comprising an integrated circuit having a processor and input/output sections are coupled in a network. Each of the programmable cells includes a unique identification number (ID) at the time it is manufactured. When organized in a network, each of the cells performs at least one of the following functions: (1) sensing, (2) communicating, (3) controlling. Groups of cells may be organized to perform group functions and may be assigned a group ID. Cells and groups may be referenced by cell ID and group ID.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AT	Austria	FR	France	ML	Mali
AU	Australia	GA	Gabon	MR	Mauritania
BB	Barbados	GB	United Kingdom	MW	Malawi
BE	Belgium	HU	Hungary	NL	Netherlands
BG	Bulgaria	IT	Italy	NO	Norway
BJ	Benin	JP	Japan	RO	Romania
BR	Brazil	KP	Democratic People's Republic of Korea	SD	Sudan
CF	Central African Republic			SE	Sweden
CG	Congo	KR	Republic of Korea	SN	Senegal
CH	Switzerland	LI	Liechtenstein	SU	Soviet Union
CM	Cameroon	LK	Sri Lanka	TD	Chad
DE	Germany, Federal Republic of	LU	Luxembourg	TG	Togo
DK	Denmark	MC	Monaco	US	United States of America
FI	Finland	MG	Madagascar		

NETWORK AND INTELLIGENT CELL FOR PROVIDING SENSING, BIDIRECTIONAL COMMUNICATIONS AND CONTROL

BACKGROUND OF THE INVENTION

1. Field of the Invention

The invention relates to the field of networks with distributed intelligence, configuration and control and intelligent cells used in networks, primarily where the networks are used for sensing, communicating and controlling.

2. Prior Art.

There are many commercially available products which provide sensing, control and communications in a network environment. These products range from very expensive, elaborate systems, to simple systems having little intelligence. As will be seen, the present invention is directed towards providing a system having a relatively large amount of intelligence and computational power but at a low cost.

One commercially available system "X-10" provides control, by way of example, between a light switch and a light. When the light switch is operated, a code pattern is transmitted over the power lines to a receiver at the light. The code pattern is transmitted twice, once in its true form and once in its complementary form. When the code is received by the receiver, it is interpreted, and thereby used to control the light. Mechanical addressing means are employed to allow the

transmitter at the switch to communicate with the specific desired receiver at the light.

As will be seen, the present invention provides substantially more capability and flexibility than current systems.

Applicant will submit prior art references on X-10 and other known prior art systems.

SUMMARY OF THE INVENTION

A network for providing sensing, communications and control is described. A plurality of intelligent cells each of which comprises an integrated circuit having a processor and input/output section are coupled to the network. Each of the programmable cells receives when manufactured a unique identification number (48 bits) which remains permanently within the cell. The cells can be coupled to different media such as power lines, twisted pair, radio frequency, infrared ultrasonic, optical coaxial, etc., to form a network.

Networks are distinguished from one another by system identification numbers (IDs). Groups of cells within each network are formed to perform particular functions and are identified by group IDs. Communications occur within the network through use of the system, group and cell IDs. Some cells (announcers) are assigned the task of sensing, for example, the condition of a switch, and others (listeners) the task of controlling, such as controlling a light. Cells can perform multiple tasks and be members of multiple groups, and, for example, can act as a repeater for one group and a listener in a another group. When manufactured, the cells are identical except for the cell ID; they

are programmed to perform specific tasks for a particular group or groups.

The preferred embodiment of the cell includes a multiprocessor and multiple I/O subsections where any of the processors can communicate with any of the I/O subsections. This permits the continual execution of a program without potential interruptions caused by interfacing with the I/O section. The I/O section includes programmable A-to-D and programmable D-to-A converters as well as other circuits for other modes of operation.

The network protocol provides great flexibility, and for instance, allows groups to be formed and/or changed after the cells are in place. As will be seen, the intelligence for the network is distributed among the cells. In general, the network is lightly loaded, although provisions are made for contention and other conditions which may arise. The communication between the cells in general is optimized for carrying out the functions assigned to groups, rather than for transmission of data unrelated to the control function of the network. For this reason, normally the packets carrying messages are relatively short compared to Ethernet, Arpa, AppleTalk, X-25 and many other broadband and data communication systems.

Other aspects of the invented network and cell will be apparent from the detailed description of the invention.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a block diagram illustrating typical application for the present invention.

Figure 2 is a diagram used to illustrate the grouping of cells.

Figure 3 is another block diagram similar to Figure 2 used to illustrate the grouping of cells.

Figure 4 is a diagram used to describe subchannels.

Figure 5 is a diagram illustrating a plurality of cells; this diagram is used to describe cell group formation employing the present invention.

Figure 6 is a chart illustrating the packet format used with the present invention.

Figure 7 is a chart illustrating the designation list portion of the packet format of Figure 6.

Figure 8 illustrates a series of steps used in forming a group of cell with the present invention.

Figure 9 is a chart illustrating the code assignments for the three-of-six encoding used with the present invention.

Figure 10 is a block diagram of the communication and control cell.

Figure 11 is a block diagram of a portion of the instruction decoding logic used within the processor of the cell of Figure 10

Figure 12 is a detailed block diagram of the process of Figure 10.

Figure 13 is a timing diagram for the processor of Figure 10; this diagram also shows latches and registers used to provide the pipelining employed by the cell.

Figure 14 is a block diagram illustrating the presently preferred embodiment of the three-of-six encoder.

Figure 15 is a block diagram showing the presently preferred embodiment of the three-of-six decoder.

Figure 16 is a block diagram showing the presently preferred embodiment of the three-of-six code verifier.

Figure 17 is an electrical schematic of the buffer section of one of the I/O sections.

Figure 18 is an electrical schematic of the counting and timing functions for an I/O subsection.

Figure 19 is an electrical schematic of the control and state machine for an I/O section.

Figure 20 is an electrical schematic for the sample and hold means associated with the I/O subsections.

Figure 21 illustrates the network formed within an I/O subsection to do digital-to-analog conversion.

Figure 22 illustrates the network formed within an I/O section for analog-to-digital conversion.

Figure 23 is an electrical schematic showing the communications portion of an I/O subsection.

Figure 24 is a state diagram used for the I/O subsections and for transmission contentions.

Figure 25 is a state diagram for the link level ARQ.

Figure 26 is a state diagram for primary station connections.

Figure 27 is a state diagram for secondary station connections.

Figure 28 is a block diagram for a grouping device.

Figure 29 is a diagram showing the form in which the system ID is encoded for transmission by the packet and encoded within a cell.

Figure 30 is a diagram used to describe the operation of the input/output section and semaphore register.

DETAILED DESCRIPTION OF THE PRESENT INVENTION

An apparatus and method for providing a communications, sensing and control in a network is described. Where the network contains a plurality of intelligent cells, the cells in general are programmable single chip remote control, sensing and communication devices that, when interconnected (via various media) with other cells, have distributed sensing, communication, control and network configuration intelligence, configuration and control. The system comprises a network of cells organized in a hierarchy based on communications needs. Cells are organized into working "groups" independent of the network hierarchy. Groups of cells generally are used to perform a group function. This function is carried out by the assignment of tasks to cells within the groups. Cells communicate, control and sense information. In general, each cell has a unique identification number and perform information processing tasks such as: bidirectional communications protocol, input/output, packet processing and analog and digital sensing and control. In general, the system comprised of the cells has the characteristic of storing network configuring information that is distributed throughout the system; and communicates automatically routed messages among cells. Each system also has a unique

identification (ID) which in the presently preferred embodiment is 48 bits. Moreover, it contains versatile programmable input/output I/O circuits with digital versatile programming to configure cells to specific sensing, communication, control and I/O, analog I/O, communication I/O and communications bit rate sensing.

In the following description, numerous specific details are set forth such as specific frequencies, etc., in order to provide a thorough understanding of the present invention. It will be obvious, however, to one skilled in the art that these details are not required to practice the invention. In other instances, well-known circuits, methods and the like are not set forth in detail in order not to unnecessarily obscure the present invention.

I.

OVERVIEW OF AN APPLICATION OF THE PRESENT INVENTION

Before describing the present invention in detail, an understanding of a typical application will aid in appreciation of the details to follow. In Figure 1, a simple, typical application is shown based on the use of the present invention in a home. In

Figure I, the switch 22 is used through the present invention to control the light 23.

The arrangement 20 comprises a cell 27 which is connected to the switch 22. The cell is also connected to a transceiver 29 which couples data onto the lines 24 and 25. Power for the transceiver and cell are provided from the power supply 30 which receives power from the lines 24 and 25. For this example, the lines 24 and 25 are ordinary household wiring (e.g., 110VAC) and the power supply 30, a five volt DC supply. The cell 27 is preferably an integrated circuit which is described in more detail beginning with Figure I0. The transceiver 29 may be any one of many well-known devices for receiving and transmitting digital data and as presently contemplated does not perform any processing on transmitted data. The entire arrangement 20 may be small enough to fit within an ordinary wallmounted electrical box which normally contains an electrical switch.

The arrangement 21 again may be small enough to fit within a typical electrical outlet box and includes a power supply 31 and transceiver 33 which may be identical in construction to the power supply 30 and transceiver 29, respectively. This cell 28 is coupled to the transceiver 30 and power supply 29 as well as the solenoid operated power switch 32. Cell 28 may be identical to

becomes for practical purposes, or no matter how many networks are interconnected. The grouping device then accesses the individual cell IDs and assigns a system ID to each cell. In addition, the grouping device configures the cells into groups to perform group related functions.

For the illustration of Figure I, cell 27 is designated as "A" to indicate that its primary function is to "announce" that is, transmit the state of switch 22 on the network communications lines 24 and 25, and 34 and 35. On the other hand, cell 28 is designated with the letter "L" since its primary function in Figure I is to "listen" to the network and in particular to listen to messages from cell 27. In subsequent figures, the "A" and "L" designations are used, particularly in connection with a group formation of multiple cells to indicate an announcer arrangement, such as arrangement 20 and a listener arrangement, such as arrangement 2I. For purposes of discussion the cells themselves are sometimes referred to as transmitting or receiving data without reference to transceivers. (In some cases, the transceivers may be a simple passive network or simple wires, which couple the input/output of a cell onto a line. As will be seen the I/O section of the cells can provide output signals that

can drive a twisted pair or the like. Thus the cells themselves can function as a transceiver for some media.)

The cells 27 and 28 as will be described subsequently are processors having multiprocessor attributes. They may be programmed prior to or after installing to perform their required function, such as an announcer or listener and for grouping combinations.

II.

NETWORK ORGANIZATION AND DEFINITIONS

A. Definitions

Cell: A cell is an intelligent, programmable element or elements providing remote control, sensing and/or communications, that when interconnected with other like elements form a communications, control and sensing network or system with distributed intelligence.

Announcer: An announcer is a source of group messages.

Listener: A listener is a sink of group messages.

(An announcer in some cases may request state information from a listener.)

Repeater: A repeater is a cell which in addition to other functions reads packets from a medium and rebroadcasts them.

Group: A set of cells which work together for a common function (for example, a switch controlling a set of lights) is referred to as a "group".

In Figure 2, the group 37 has an announcer 37a, listeners 37b, and 37c, and a listener 40. A group 38 includes an announcer 38a, listeners 38b and 38c and the listener 40. Figure 2 illustrates that a single cell (cell 40) may be a listener in two

groups. If announcer 37a has a light switch function, it can control lights through cells 37b, 37c and 40. Similarly, a switch associated with announcer 38a can control lights through cells 37c, 37b, and 40.

In Figure 3, a group 42 includes announcers 44, 45 and listeners 46 and 47. The group 43 shares cell 44 with group 42; however, cell 44 is a listener for group 43. The group 41 shares cell 47 with group 42; cell 47 is an announcer for group 41 and for example, can announce to the listener 48 of group 41. Cell 47 also operates as a listener for group 42. A single cell as shown may be an announcer for one group and a listener for another group (cells are programmed to perform these functions, as will be discussed). However, as presently contemplated, a single cell cannot announce for more than one group.

(In the currently preferred embodiment each cell has three input/output pairs of lines and a select line. Each pair shares a common set of resources. The lines may be used independently for some functions where the required shared resources do not conflict. In other functions, the lines are used as pairs. In this example, a pair of leads from cell 27 are coupled to a light switch and another pair are used for communications from the announcer, cell 27.)

Subchannel: In Figure 4, a first plurality of cells are shown communicating through a common medium such as a twisted pair 50 (cells are shown as "C", announcers as "A" and listeners as "L"). This (e.g., twisted pair 50) is defined as a subchannel, that is, a set of cells all of which communicate directly with one another over the same medium. A broadcast by any member of the subchannel, such as the cell 49, will be heard by all members of that subchannel over the twisted pair 50.

Channel: A channel comprises two or more subchannels where all the cells communicate using the same medium. In Figure 4, another plurality of cells are shown coupled to twisted pair 52 forming another subchannel. Assume cells 56 and 57 communicate between one another through a twisted pair 72. They form yet another subchannel. The cells associated with the twisted pairs 50, 52 and 72 comprise a single channel. It is possible that the twisted pairs 50, 52 and 72 are one continuous twisted pair with one subchannel 50 so far apart from the second subchannel 52 that the only communications between subchannels is over the portion of the twisted pair 72 running between cells 56 and 57. In this case the cells 56 and 57 are assigned to be "repeaters" in addition to whatever other function they may serve (e.g., announcer or listener).

A group 55 is illustrated in Figure 4 which comprises an announcer and listener in the two different subchannels. Another group 75 is illustrated comprising an announcer on one subchannel 51 and subchannel 52, where the subchannels are not part of the same channels since they use different media.

Gateway: A gateway reads packets from two different media and rebroadcasts them. A cell may be a gateway. Communications between channels is through gateway 54.

In Figure 4, an additional subchannel which includes the cell 58 is coupled to another medium 5l, for example, a common power line. The cell 58 is shown connected to channel gateway 54 which in turn communicates with the twisted pair 52. The gateway 54 does not necessarily perform either an announcer or listener function, but rather for the illustrated embodiment, performs only a channel gateway function by providing communication between two different media.

Subnetwork: A subnetwork comprises all the cells having the same system identification (system ID). For example, all the cells in a single family home may have the same system ID. Therefore, the channels of Figure 4 may be part of the same subnetwork in that they share the same system ID.

Full Network: A full network may comprise a plurality of subnetworks each of which has a different system ID; a communications processor is used for exchanging packets between subnetworks. The communications processor translates packets changing their system ID, addressing and other information. Two factory buildings may each have their own system ID, but control between the two is used by changing system IDs. (The word "network" is used in this application in its more general sense and therefore refers to other than a "full network" as defined in this paragraph.)

Other terms used later are:

Probe Packet: A packet routed by flooding which accumulates routing information as it travels through the network.

Grouping Device: A device that controls determination of routes among cells, assigns cells to groups, and assigns function to group members.

Contention: The state which exists when two or more cells attempt to transmit a broadcast on the same subchannel at the same time and their signals interfere.

B. GROUP FORMATION

1. Cells Assigned to a group by a postinstallation grouping device.

Assume that the plurality of cells shown in Figure 5 are all connected to communicate over the power lines in a home and are part of the same channel. Further assume that one cell, announcer 60, is to be grouped with the listener 65. The lines between the cells such as line 59 is used to indicate which of the cells can communicate directly with one another, for instance, announcer 60 and cell 61 can communicate with one another. (Cells 61, 62, 63, 64 and 66 of course may be announcers or listeners in other groups, but for purposes of explanation are shown as "C" in Figure 5.) Since announcer 60 and cells 61, 62, and 63 all communicate with one another, they are on the same subchannel. Similarly, cells 62, 64, 65 and 66 are another subchannel. (There are other subchannels in Figure 5). Importantly, however, announcer 60 and listener 65 are in different subchannels of the channel of Figure 5 and there are numerous routes by which a message can be passed from announcer 60 to listener 65, for example, through cells 61 and 64 or through cells 62 and 64, etc.

Note that even though all the cells are on the same power system of a house, they may not communicate directly with one

another. For instance, the announcer 60 may be on one circuit which is only coupled to the listener 65 through long lengths of wire running the length of a home and a low impedance bus bar of a circuit breaker panel. The high frequency communication messages may be sufficiently attenuated through this path to prevent direct communications between cells even though they are physically close to one another.

For the following description, it is assumed that each of the cells can broadcast without interfering with the broadcast of other cells. That is, messages do not interfere with one another. The case where some contention occurs is dealt with under the protocol section of this application.

In one embodiment, the group of announcer 60 and listener 65 is formed by using the grouping device shown in Figure 28. Note that before this group is formed the announcer 60 and listener 65 are ordinary cells, not designated to be an announcer and listener. Each grouping device may be assigned a unique 48 bit system ID at time of manufacture (in the presently preferred embodiment a 48 bit number is used). In the presently preferred embodiment, a cell is included with each grouping device. The cell's ID becomes the system ID. This assures that each system has a unique system ID. By way of example, each home has its

own "grouping" device and hence, its own system ID for the subnetworks used in the home. This system ID is used in cell packets for the subnetwork. In this example, the grouping device has available the cell IDs of cells 60 and 65. (Various methods of obtaining cell IDs will be described later.)

The grouping device is connected to cell 60 by communicating through one of its three pairs of input/output (I/O) lines of the cell (or the select pin) and the grouping device reads the 48 bit ID number of the cell 60. (Different methods of determining the cell's IDs are described in the next section.) The grouping device next generates a random bit binary number which in the presently preferred embodiment is 10 bits. This number functions as a group identification number (also referred to as the group address) for the group comprising the announcer 60 and listener 65. The grouping device checks this number against other group IDs which it has previously assigned to determine if the group ID has previously been used. If it has been already used it generates another number. (A single grouping device, for instance keeps track of all the group IDs assigned in a single home.) The grouping device programs the cell 60 designating it as an announcer.

The grouping device may cause the announcer 60 to broadcast the group number in a special packet which asks all cells in the network to acknowledge the message if they have been designated as a member of this group. This is another way to verify that the group ID has not been used.

The grouping device now determines the ID number of the cell 65. This may be done by connecting the grouping device directly to the cell 65 even before the cell is installed or by other methods discussed in the next section. (A cell and a group can be assigned ASCII names, for example, "porchlight" (cell name) and "exterior lights" (group name). This is used to allow selection of cell IDs or group IDs by accessing the ASCII name.

Now the grouping device causes the announcer 60 to transmit a probe packet. The probe packet contains the ID of cell 65. The packet directs all cells receiving the packet to repeat it and directs cell 65 to acknowledge the packet. Each cell receiving the probe packet repeats it and adds to the repeated packet its own ID number. Each cell only repeats the packet once (the mechanism for preventing a probe packet from being repeated more than once is described later.)

The cell 65 receives the probe packet through numerous routes, including those which in the diagram appear to be most

direct (via cell 62) and those which are longer, for example, via cells 61 and 64. It is assumed that the first probe packet to arrive at cell 65 took the most direct route and is therefore the preferred routing. (Assume that this is via cell 62.) Cell 65 receives a packet which indicates that the probe packet was transmitted by cell 60, repeated by cell 62 and intended for cell 65. The other probe packets received by cell 65 after this first packet are discarded by cell 65.

Cell 65 now transmits an acknowledgement back to announcer 60. This packet includes the routing of the probe packet (e.g., repeated by cell 62). The packet directs cell 62 to repeat the packet to confirm its receipt.

After announcer 60 receives the acknowledgement packet for cell 65 it determines that cell 62 must be a repeater. The grouping devices causes announcer 60 to send a repeater assignment packet which includes the unique ID number of cell 62, the group number and a message which informs cell 62 that it is assigned a repeater function for the group. This causes cell 62 to repeat all those packets for the group comprising announcer cell 60 and 65. Another message is sent from announcer 60 under control of the grouping device repeated by cell 62, designating cell 65 as a listener, causing it to act upon messages for the

group (cell 65 becomes a group member.) The grouping device assigns members a member number whch is stored by member cells.

The group formation described above is shown in Figure 8 by steps or blocks 68 through 72. Block 68 illustrates the broadcasting of the probe packet (e.g., cell 60 transmits the initial probe packet to all cells). The packet includes the address of a destination cell. As the packet proceeds through the network, the packet and accumulates the ID numbers of those cells repeating the packet (block 69). Block 70 shows the acknowledgement (reply) to the probe packet from the destination cell (e.g., cell 65). This packet returns the ID numbers of the repeaters contained in the first received probe packet. Repeater assignment packets are sent out by the announcer causing each repeater to rebroadcast packets for the group; this is shown by block 71. Finally, as shown by block 72, the destination cell such as cell 65 is designated as a listener.

2. Cells assigned to a group by a preinstallation grouping device.

There may be several types of preinstallation grouping devices, for example, see Figure 28 for a device which may be used. One type is a device that a manufacturer uses to preassign cells to groups. Another type of preinstallation grouping device

is one that a retailer or other cell vendor may use to assign cells to groups before installation.

A grouping device assigns a cell to a group and assigns the cell's function(s) for that group. The grouping device may also assign a system ID to the cell. The system ID assigned by a preinstallation grouping device is not necessarily a unique system ID. (Postinstallation grouping devices assign a unique system ID to each system.)

One method that may be used by preinstallation grouping devices to generate a system ID is to choose a system ID from a range of the 48 bit address and system ID numbers that have been set aside for use as preinstallation system IDs. Just as the cell IDs in the range 1-1023 have been set aside for use as group IDs and group addresses, the cell IDs in the range 1024-2047 can be set aside for use as preinstallation system IDs.

It is desirable that grouping devices and other network control devices be able to identify preinstallation system IDs as opposed to postinstallation system IDs. Since postinstallation sytem IDs are generated by copying a cell ID, cell IDs should not be assigned in the range set aside for preinstallation system IDs. Therefore, ID numbers in that range would not be assigned to cells as cell IDs.

Cells may be sold in sets that have been preassigned to a group by the manufacturer. The type of preinstallation grouping device used by the manufacturer assigns cells to groups by writing the appropriate codes into the cells' nonvolatile memory. The user may install such a set of cells and it will operate without assignment by a postinstallation grouping device provided that the set of cells may communicate via a single subchannel.

A user may assign cells to a group at the time cells are purchased or at any other time before installation. Such cells, unlike the case previously discussed, are not assigned to groups by the manufacturer and are called unassigned cells. Unassigned cells all have the same system ID, a system ID number that has been set aside for use only by unassigned cells.

The user assigns a set of cells to a group by using a preinstallation grouping device that may be different from the preinstallation grouping device used by a manufacturer.

Typically, such a grouping device will operate on one cell at a time. The operator commands the grouping device to generate a new group ID and system ID and then each cell is connected to the device in turn. The operator commands the grouping device to assign a cell to the group while the cell is connected to the

grouping device. The grouping device assigns cells the same group ID and system ID until it is commanded by the operator to generate a new group ID and system ID.

The user may install such a set of cells and it will operate without use of a postinstallation grouping device provided that the set of cells can communicate via a single subchannel.

3. Unassigned Cells Grouping and Self-Assignment After Installation.

Unassigned cells may create a group and assign themselves to the group after installation in the following manner.

The first announcer cell that is stimulated via its sensor input (e.g., light switch) controls the group formation process. It chooses a system ID number at random from the range of system ID numbers that have been set aside for preinstallation grouping devices. It chooses a group ID number at random. It then broadcasts the group ID number in a packet that requests a reply from any cells that are members of that group. If the transmitting cell receives any such replies, it chooses another group ID at random. The cell continues this process of selecting a random group ID and testing to see if it is already in use until it finds a group ID that is unused in the system in which it is operating.

An unassigned cell's default configuration information programmed at the factory identifies its function as either a listener or an announcer. If the unassigned cell is an announcer, it waits for its sensing input to be stimulated, and when it is stimulated, the cell transmits a packet addressed to a group.

If an unassigned cell is a listener, it listens after power-up for a packet. The cell takes the group ID from the first packet it receives and assigns itself to that group. The cell then sends a reply to the announcer cell. This reply is not an acknowledgement only packet; it is a packet that identifies the cell as a listener in the group and the packet must be acknowledged by the announcer. This assures that all of the listener identification packets will arrive at the announcer even though there will be contention and collisions in the process.

The cell that transmitted the group announcement builds a list of group members as each reply comes in. It then sends a packet to each listener assigning that listener a group member number.

4. Unassigned Cells Joining Preexisting Group After Installation.

Unassigned cells may be added to existing systems and assigned to a group in a manner similar to the above method

discussed in Section 3 above. A listener joins the system and a group by the same method as in Section 3 above.

In the above example, the announcer waits to be stimulated via its sensor input. An unassigned announcer waits for its first sensor input stimulation or its first received packet. Of those two events, the event that occurs first determines the subsequent actions of the announcer cell.

If the cell is stimulated first, it controls a group formation process just as in the above example. If the announcer cell receives a group packet first, it joins that group as an announcer. It then sends a packet to the group announcer requesting configuration information about the group (group size, number of announcers,etc.) and the assignment of a group member number.

C. METHODS OF IDENTIFYING A CELL FOR GROUPING

In order for a grouping device to go through the steps necessary to form a group or add a cell to a group, it must know the IDs of the cells to be added to the group. The grouping device then uses those cell IDs to address commands to the cells during the grouping process. The methods that a user with a grouping device may use to obtain the cell IDs are listed below. Note that a grouping device or other control device's ability to communicate with a cell in the following example may be limited by security procedures if used. The security procedures, limitations on communications and levels of security are not critical to the present invention. The following example assumes that no security procedures are in place. In particular, it may be impossible for a grouping device to communicate with installed cells unless the grouping device has the system key (system ID and encryption keys.)

1. Direct connection to the cell.

The grouping device may be connected to an I/O line of the cell package and then send a message to the cell requesting its ID. Physical connection can be used to find a cell's ID either before or after the cell is installed. Known means can be used (e.g., a fuse

or a programmed disable command) to allow a user to disable this function in an installed cell to protect the security of the system.

2. Selection of the Cell Through Use of Special Pin

The user may use the grouping device or some other selection device to physically select the cell by stimulating a cell input pin that has been designated to serve the selection function. The grouping device communicates with the cell through the normal communications channels and sends a broadcast message requesting that all selected cells reply with their ID. Only one cell is selected so only that cell will reply to the request. Physical selection can be used to find a cell's ID either before or after the cell is installed. Again, a means can be provided to allow a user to disable this feature to protect the security of the system.

3. Query All Names of Previously Grouped Cells

It is assumed in this example that ASCII "groups" and "cell" names have been previously assigned to the cells. For this method, the grouping device queries all of the cells in a system to report their group and cell names (ASCII name). The user scrolls through the list of group names by using the grouping device. The user selects the name of the group that is believed to contain the target cell. The grouping device displays the names of all of the

cells that are in the group and their assigned tasks (announcer, listener, repeater). The user selects the name of the cell that is believed to be the target cell.

If the selected cell is an announcer, the grouping device prompts the user to activate the announcer by stimulating its input. For example; if the cell is attached to a light switch, the user turns the switch on and off. The cell sends announcement packets to the group. The grouping device listens to the communications channel and discovers the group and member numbers or other codes of the activated announcer.

If the selected cell is a listener cell, the grouping device sends packets to the cell (using the group and member numbers for addressing) commanding it to toggle its output. For example, if the cell controls a light, the light will flash on and off. This allows the user to verify that he has selected the correct cell.

The grouping device sends a packet (using group and member numbers for addressing) to the target cell with a command for the target cell to return its cell ID. The grouping device now knows the target ID and can proceed with the group assignment process.

Querying names is used to find a cell's ID before or after the cell is installed.

4. Stimulate Group.

This method is used in a network in which group and cell ASCII names have been assigned. The user commands the grouping device to wait for the next group announcement. Then the user stimulates the announcer in the group of interest. For example, if the announcer is a light switch, the user throws the switch. The grouping device hears the announcement packet and extracts the group ID from it.

The user may verify that this group ID is for the desired group by causing the grouping device to send packets to all of the group listeners commanding them to toggle their outputs. The user verifies that it is the desired group by observing the actions of the listener cells (for example, if the group consists of lighting controls, the light flashes).

Now using that group ID, the grouping device broadcasts a packet to the group requesting that each cell reply with its cell name until the cell of interest is found. The user selects that name and the grouping device, knowing that cell's ID, can proceed with the group assignment process.

If a user elects, the ID of the cell may be verified before proceeding with the grouping procedure. The following procedure is used to verify that the ID is for the target cell.

If the selected cell is an announcer, the grouping device prompts the user to activate the announcer by stimulating its input. For example: if the cell is attached to a light switch, the user turns the switch on and off. The grouping device is then able to discover the group address and member number of the cell.

If the selected cell is a listener, the grouping device sends packets to the cell (using the group and member numbers, for addressing) commanding it to toggle its output. For example, if the cell controls a light, the light will flash on and off. This allows the user to verify that he has selected the correct cell.

5. Stimulate Announcer.

This method is used in a network in which no group or cell ASCII names have been assigned but announcers and listeners have been assigned. The grouping device sends a packet to all cells in the network commanding each announcer to broadcast a packet containing its ID the next time it is stimulated. The grouping device then prompts the user to stimulate the announcer by activating its sensed device; for instance, turn on a light switch for a light switch announcer. Since the user will stimulate only one announcer, the grouping device will receive only one packet with a cell ID.

There is a chance that another announcer cell will be stimulated at the same time. Perhaps someone else throws a light switch or a temperature sensor detects a temperature change. The user may want to verify that the ID received is for the correct cell. To verify that the cell ID is the correct one, the user goes through the announcer stimulation process a second time and verifies that the same results occur.

6. **Toggle Listener**

This method is used in a network in which no group or cell names have been assigned. The grouping device broadcasts a packet that queries cells that are listeners to reply with their ID. The grouping device needs to limit the number of cells replying so the packet contains an ID bit mask to limit replies to a subset of the possible cell IDs. When the grouping device has developed a list of listener IDs, it allows the user to toggle each listener, causing the listener cell to turn its output on and off. The user continues through the list of listener cells until he observes the target cell toggling its output. The user has then identified the cell to the grouping device and it can proceed with the grouping operation.

D. PACKET FORMAT

Each packet transmitted by a cell contains numerous fields. For example, a format used for group announcements is shown in Figure 6. Other packet formats are set forth in Appendix A. Each packet begins with a preamble used for synchronizing the receiving cells' input circuitry (bit synch). The particular preamble code used in the currently preferred embodiment is described as part of the three-of-six combinatorial codes (Figure 9). A flag field of 6 bits begins and ends each of the packets. The flag field code is also described in Figure 9.

As currently preferred, each of the cells reads-in the entire packet, does a cyclic redundancy code (CRC) calculation on the packet except for the contention timer field and compares that result with the CRC field of the received packet. The ALU 102 of Figure 12 has hardware for calculating the packet CRC and CRC registers 130 for storing intermediate results. If the packet CRC cannot be verified for an incoming packet, the packet is discarded. The packet CRC field is 16 bits as calculated, then converted into 24 bit fields for transmission in a 3-of-6 code using the encoding of Figure 9. (For the remainder of discussion of packet fields in this section, the field length is described prior to encoding with the 3-of-6 combinatorial codes of Figure 9.) In

the currently preferred embodiment the CRC is a CCITT standard algorithm

$(X^{16}+X^{12}+X^{5}+1)$.

The system ID is a 32 bit field as currently preferred. The other 16 bits of the 48 bit system ID are included in the CRC calculation but not transmitted as part f the packet (Figure 29).

The link address field is a 48 bit field. When this field is all zeroes the packet is interpreted as a system wide broadcast which is acted upon by all the cells. For instance, a probe packet has an all zero field for the link address. Group addresses are contained within the link address. For group addresses the first 38 bits are zero and the remaining 10 bits contain the group address. (The cell ID numbers assigned at the factory mentioned earlier range from 1024 to 2^{48} since 2^{10} addresses are reserved for groups.) The link address, in some cases, is an individual's cell's address. (For example, when a cell is being assigned the task of repeater or listener.)

The contention timer is a 10 bit field with an additional 6 bits for a CRC field (or other check sum) used to verify the 10 bits of the timer field. Each cell which repeats a packet operates upon this field if the cell must wait to transmit the packet. If packets are being transmitted by other cells a cell must wait to

transmit its packet, the time it waits is indicated by counting down the contention timer field. The rate at which this field is counted down can be programmed in a cell and this rate is a function of the type of network. The field starts with a constant which may be selected by the type of network. Each cell repeating the packet counts down from the number in the field at the time the packet is received. Therefore, if a packet is repeated four times and if each of the four cells involved wait for transmitting, the number in the contention field reflects the sum of the times waited subtracted from a constant (e.g., all ones). When the contention timer field reaches all zeroes, the cell waiting to transmit the packet discards the packet rather than transmit it. This prevents older packets from arriving and being interpreted as being a new packet.

As mentioned, the contention timer has its own 6 bit CRC field. If the contention timer field were included in the packet CRC, the packet CRC could not be computed unfil a packet could actually be transmitted. This would require many calculations in the last few microseconds before a transmission. To avoid this problem a separate CRC field is used for the contention timer field. If the contention timer field cannot be verified by its 6-bit CRC, the packet is discarded.

The hop count field records the number of hops or retransmissions that a packet takes before arriving at its destination. This 4 bit field starts with a number which is the maximum number of retransmissions allowed for a particular packet and is decremented by each cell repeating a packet. For example, in a packet originated by a group announcer the starting "hop" count is the maximum number of retransmissions that the packet must undergo to reach all of the cells in a group. When this field becomes all zeroes, the packet is discarded by the cell, rather than being retransmitted Therefore, 16 hops or retransmissions is the limit as currently implemented.

The link control field provides the link protocol and consists of 8 bits. This field is discussed in a subsequent section covering other layers of the protocol.

The randcm/pseudo random number field contains an 8 bit random number which is generated for each packet by the cell originally transmitting the packet. This number is not regenerated when a packet is repeated. This number is used as will be explained in conjunction with Figure 8 to limit rebroadcasting of probe packets; it also may be used in conjunction with encryption where the entire packet is to be encrypted.

The network control field (4 bits) indicates routing type or packet type, for instance, network control, group message, probe message, etc.

The source address field (variable size) contains, by way of example, the 48 bit ID number of the cell originating a packet. For a probe packet this field contains the ID number of the announcer. For an acknowledgement the field contains the ID of the listener. For a packet addressed to a group, this field contains the source cell's group member number.

The destination list is described in conjunction with Figure 7.

The message field is variable in length and contains the particular message being transmitted by the packet. Typical messages are contained in Appendix B. In the case of a probe packet the field includes the routing; that is, each cell repeating includes its ID number to this field. The messages, once a group is formed, will, for instance, is used by announcer 60 to tell listener 65 to turn-on a light, etc.

The encryption field, when used, contains 16 bits used to verify the authenticity of an encrypted packet typically this portion of a packet is not changed when a packet is repeated. Well-known encryption techniques may be used.

The bracket 99 of Figure 6 represents the portion of a packet which remains unaltered when a packet is repeated. These fields are used to limit repeating as will be described in conjunction with Figure 8.

The destination list field of the packet of Figure 6 is shown in Figure 7. The destination list begins with a 4 bit field which indicates the number of members in a group designated to receive a message in the packet. Therefore the packet can be directed to up to 16 members of a group. The number of each of the members within the group is then transmitted in subsequent 8 bit fields. The group number contained in the link address and member number contained in the destination list forms an address used to convey messages once the group is formed. If the destination number is zero, the packet is addressed to all members of the group. For some packet types this field contains the ID of the receiving cell (see Appendix A).

E. MECHANISM FOR PREVENTING REBROADCASTING OF CERTAIN PACKETS

As previously mentioned, the probe packets are repeated only once by each of the cells after the packet is initially

broadcast. A special mechanism programmed into each of the cells allows the cells to recognize packets which it has recently repeated.

First, it should be recalled that as each cell transmits, or retransmits a packet, it calculates a packet CRC field which precedes the end flag. For packets that are repeated, a new CRC is needed since at least the hop count will change, requiring a new packet CRC field for the packet. This CRC field is different from the CRC field discussed in the next paragraph.

As each packet requiring repeating is received, a repeater CRC number is calculated for the fields extending from the beginning of the link control to the end of the destination list as indicated by bracket 99 of Figure 6. As a cell rebroadcasts a packet it stores the 16 bit repeater CRC results in a circular list of such numbers if the same number is not already stored. However, the packet is repeated only if the circular list does not contain the repeater CRC results calculated for the field 99.

Therefore, as each packet is received which requires repeating, the CRC is computed for the field 99. This is shown by block 73a of Figure 8. This number is compared with a list of 8 numbers stored within the RAM contained within the cell indicated by block 73b. If the number is not found within the

stored numbers, the new repeater CRC results are stored as indicated by block 73c and the packet is repeated. On the other hand, if the number is found then the packet is not repeated. As presently implemented, 8 numbers are stored in a circular list, that is, the oldest numbers are discarded as new ones are computed.

The use of the repeater CRC calculation associated with the field 99 and the use of the circular list will prevent repeating of a previously rebroadcasted packet. Note that even if an announcer continually rebroadcasts the same sequence of messages, for example, as would occur with the continuous turning on and turning off of a light, a cell designated as a repeater will rebroadcast the same message since the packet containing messages appears to be different. This is true because the random number sent with each of the identical messages will presumably be different. However, in the instance where a cell receives the same message included within the same field 99 (same random number), the packet with its message will not be rebroadcast. This is particularly true for probe packets. Thus, for the establishment of groups discussed above, the broadcast probe packets quickly "die out" in the network, otherwise they may echo

for some period of time, causing unnecessary traffic in the network.

F. THREE-OF-SIX-COMBINATORIAL CODING

In many networks using the synchronous transmission of digital data, encoding is employed to embed timing information within the data stream. One widely used encoding method is Manchester coding. Manchester or other coding may be used to encode the packets described above, however, the coding described below is presently preferred.

A three-of-six combinatorial coding is used to encode data for transmission in the presently preferred embodiment. All data is grouped into 4 bit nibbles and for each such nibble, six bits are transmitted. These six bits always have three ones and three zeroes. The transmission of three ones and three zeroes in some combination in every six bits allows the input circuitry of the cells to quickly become synchronized (bit synch) and to become byte synchronized as will be discussed in connection with the I/O section. Also once synchronized (out of hunt mode) the transitions in the incoming bit stream are used to maintain synch.

The righthand column of Figure 9 lists the 20 possible combinations of 6 bit patterns where 3 of the bits are ones and 3 are zeroes. In the lefthand column, the corresponding 4 bit

pattern assigned to the three-of-six pattern is shown. For example, if the cell is to transmit the nibble 0111, it is converted to the bit segment 010011 before being transmitted. Similarly, 0000 is converted to 011010 before being transmitted. When a cell receives the 6 bit patterns, it converts them back to the corresponding 4 bit patterns.

There are 20 three-of-six patterns and only 16 possible 4 bit combinations. Therefore, four three-of-six patterns do not have corresponding 4 bit pattern assignments. The three-of-six pattern 010101 is used as a preamble for all packets. The flags for all packets are 101010. The preamble and flag patterns are particularly good for use by the input circuitry to establish data synchronization since they have repeated transitions at the basic data rate. The two three-of-six patterns not assigned can be used for special conditions and instructions.

Accordingly, a cell prepares a packet generally in integral number of bytes and each nibble is assigned a 6 bit pattern before transmission. The preamble and flags are then added. The circuitry for converting from the 4 bit pattern to the 6 bit patterns and conversely, for converting from the 6 bit patterns to the 4 bit patterns is shown in Figures 14 and 15.

III.

COMMUNICATION AND CONTROL CELL

A. Overview of the Cell

Referring to Figure 10, each cell includes a multiprocessor 100, input/output section 107-110, memory 115 and associated timing circuits shown specifically as oscillator 112, and timing generator 111. Also shown is a voltage pump 116 used with the memory 115. This cell is realized with ordinary integrated circuits. By way of example, the multiprocessor 100 may be fabricated using gate array technology, such as described in U.S. Patent 4,642,487. The preferred embodiment of the cell comprises the use of CMOS technology where the entire cell of Figure 10 is fabricated on a single silicon substrate as an integrated circuit. (The multiprocessor 100 is sometimes referred to in the singular, even though, as will be described, it is a multiprocessor, specifically four processors.)

The multiprocessor 100 is a stack oriented processor having four sets of registers 101, providing inputs to an arithmetic logic unit (ALU) 102. The ALU 102 comprises two separate ALU's in the presently preferred embodiment.

The memory 115 provides storage for a total of 64KB in the currently preferred embodiment, although this particular size is not critical. One portion of the memory is used for storing instructions (ROM code 115a). The next portion of the memory is a random-access memory 115b which comprises a plurality of ordinary static memory cells (dynamic cells can be used). The third portion of the memory comprises an electrically erasable and electrically programmable read-only memory (EEPROM) 115c. In the currently preferred embodiment, the EEPROM 115c employs memory devices having floating gates. These devices require a higher voltage (higher than the normal operating voltage) for programming and erasing. This higher potential is provided from an "on-chip" voltage pump 116. The entire address space for memory 115 addressed through the ALU 102a which is one part of the ALU 102.

The ROM 115a stores the routines used to implement the various layers of the protocol discussed in this application. This ROM also stores routines needed for programming the EPROM 115c. The application program for the cell is stored in ROM 115a and, in general, is a routine which acts as a "state machine" driven by variables in the EEPROM 115c and RAM 115b. RAM 115b stores communications variables and messages, applications

variables and "state machine" descriptors. The cell ID, system ID and communications and application parameters (e.g., group number, member number, announcer/repeater/listener assignments) are stored in the EEPROM 115c. The portion of the EEPROM 115c storing the cell ID is "write-protected" that is, once programmed with the cell ID, it cannot be reprogrammed

The input/output section of the cell comprises four subsections 107, 108, 109 and 110. Three of these subsections 107, 108 and 109 have leads 103, 104, and 105 respectively for communicating with a network and/or controlling and sensing devices connected to the cell. The remaining subsection 110 has a single select pin 106 which can be used to read in commands such as used to determine the cell's ID. As presently implemented, the subsection 110 is primarily used for timing and counting. The input/output section is addressed by the processor through a dedicated address space, and hence, in effect appears to the processor as memory space. Each I/O subsection can be coupled to each of the subprocessors. This feature, along with the multiprocessor architecture of processor 100, provides for the continuous (non-interrupted) operation of the processor. The I/O section may be fabricated from well-known circuitry; the

presently preferred embodiment is shown in Figures 17 through 23.

The cell of Figure 10 also includes an oscillator 112 and timing generator 111, the latter provides the timing signals particularly needed for the pipelining shown in Figure 13. Operation at a 16mHz rate for the phases 1-4 of Figure 13 is currently preferred, thus providing a 4mHz minor instruction cycle rate. Other well-known lines associated with the cell of Figure 10 are not shown (e.g., power).

All of the cell elements associated with Figure 10 are, in the preferred embodiment, incorporated on a single semiconductor chip, as mentioned.

B. PROCESSOR

The currently preferred embodiment of the processor 100 is shown in Figure 12 and includes a plurality of registers which communicate with two ALU's 102a and 102b. (Other processor architectures may be used such as one having a "register" based system, as well as other ALU and memory arrangements.) The address ALU 102a provides addresses for the memory 115 and for accessing the I/O subsections. The data ALU 102b provides data for the memory and I/O section. The memory output in general is

coupled to the processor registers through registers 146 to DBUS 223.

The 16-bit ABUS 220 provides one input to the address ALU 102a. The base pointer registers 118, effective address registers 119 and the instruction pointer registers 120 are coupled to this bus. (In the lower righthand corner of the symbols used to designate these registers, there is shown an arrow with a designation "x4". This is used to indicate that, for example, the base pointer register is 4 deep, more specifically, the base pointer register comprises 4 16-bit registers, one for each processor. This is also true for the effective address registers and the instruction pointer registers.) The BBUS 221 provides up to a 12 bit input to the ALU 102a or an 8 bit input to the data ALU 102b through register 142. The 4 deep top of stack registers 122, stack pointer registers 123, return pointer registers 124 and instruction registers 125 are coupled to the BBUS.

The CBUS 222 provides the other 8-bit input to the ALU 102 through register 143. The CBUS is coupled to the instruction pointer registers 120, the 4 deep top of stack registers 122, the four carry flags 129, and the 4 deep CRC registers 130 and the 4 deep next registers 131.

The MBUS, coupled to the output of the memory, can receive data from the output of the ALU 102b through register 145b, or from the memory or I/O sections (107-110). This bus through register 146 and the DBUS 223 provides inputs to registers 118, 119, 120, 122, 123, 124, 125, 130, 131 and to the carry flags 129.

There is a 16-bit path 132 from the output of the address ALU 102a to the registers 120. The ALU 102b includes circuitry for performing CRC calculations. This circuitry directly connects with the CRC registers 130 over the bidirectional lines 133. The top of stack registers 122 are connected to the next registers 131 over lines 138. These lines allow the contents of register 122 to be moved into registers 131 or the contents of register 131 to be moved into registers 122. As currently implemented, a bidirectional, (simultaneous) swap of data between these registers is not implemented. Four bits of data from the output of the memory may be returned directly either to the instruction pointer registers 120 or the instruction registers 125 through lines 139.

The pipelining (registers 141,142, 143, 145 and 146) of data and addresses between the registers, ALU, memory and their respective buses is described in conjunction with Figure 13.

The data in any one of the stack pointer registers 123 or any one of the return pointer registers 124 may be directly incremented or decremented through circuit 127.

Both ALU's 102a and 102b can pass either of their inputs to their output terminals, can increment and can add their inputs. ALU 102b in addition to adding, provides subtracting, shifting, sets carry flags 124 (when appropriate), ANDing, ORing, exclusive ORing and ones complement arithmetic. The ALU 102b in a single step also can combine the contents of next registers 131 and CRC registers 130 (through paths 222 and 133) and combine it with the contents of one of the top of stack registers 122 to provide the next number used in the CRC calculations. Additionally, ALU 102b performs standard shifting and provides a special nibble feature allowing the lower or higher four bits to be shifted to a higher or lower four bits, respectively. Also, ALU 102b performs a 3-of-6 encoding or decoding described in Section F.

In the preferred embodiment with a single semiconductor chip for a cell there are basic contact pads on the die for power and ground and all the I/O pins A and B and the "read only" pin 106 (subsections 107, 108, 109 and 110, Figure 12). These contact

pads are used for attachment to package pins for a basic inexpensive package.

In addition to the basic contact pads additional pads in the presently preferred embodiment will be provided with connections to the ADBUS 224 and the MBUS 225 of Figure 12. One control contact pad may be provided to disable internal memory. By activating the control contact the internal memory is disabled and the data over ADBUS and MBUS is used by the processors. This allows the use of a memory that is external to the cell. It is assumed that the additional contact pads may not be available for use when the cell is in an inexpensive package. These additional contacts may be accessed by wafer probe contacts or from pins in packages that have more than the minimum number of pins.

The cell as manufactured requires an initialization program. At wafer probe time the external memory is used for several purpose, one or which is to test the cell. Another use is to provide a program to write the cell ID into the EEPROM during the manufacturing process. Any necessary EEPROM instructions to allow power up boot when the cell is later put in use may be added at this time. Initialization programs and test programs are well-known in the art.

C. PROCESSOR OPERATION

In general, memory fetches occur when the ALU 102a provides a memory address. The memory address is typically a base address or the like on the ABUS from one of the base points in registers 118, effective address registers 119 or instruction pointer register 120 combined with an offset on the BBUS from the stack pointer register 123, return pointer register 124, top of stack registers 122 or the instruction registers 125.

Calculations in the ALU 102b most typically involve one of the top of stack registers 122 (BBUS) and the next registers 131 (CBUS) or data which may be part of an instruction from one of the instruction registers 125.

While in the presently preferred embodiment, the processor operates with the output of the memory being coupled to the DBUS 223 through register 146, the processor could also be implemented with data being coupled directly to the input of ALU 102b. Also, the function performed by some of the other registers, such as the effective address registers 119 can be performed by other registers, although the use of the effective address registers, and for example, the CRC registers, improve the operation of the processor.

In general, for memory addressing, a base pointer is provided by one of the registers 118, 119 or 120 with an offset from one of registers 122, 123, 124 or 125. The address ALU 120a provides these addresses. Also, in general, the ALU 120b operates on the contents of the top of stack and next register; there are exceptions, for example, the instruction register may provide an immediate input to the ALU 102b. Specific addressing and other instructions are described below.

D. MULTIPROCESSOR OPERATION

The processor is effectively a multiprocessor (four processors) because of the multiple registers and the pipelining which will be described in conjunction with Figure I3. As mentioned, one advantage to this multiprocessor operation is that interrupts are not needed, particularly for dealing with input and output signals. The multiprocessor operation is achieved without the use of separate ALUs for each processor. In the currently preferred embodiment, economies of layout are obtained by using two ALUs, (I02a and I02b) however, only one of the ALUs operates at any given time. (Note the BBUS provides an input to both ALUs.) Therefore, the multiprocessor operation of the present invention may be obtained using a single ALU.

The processing system has four processors sharing an address ALU, a data ALU and memory. A basic minor cycle takes four clock cycles for each processor. The ALUs take one clock cycle and the memory takes one clock cycle. The minor cycles for each processor are offset by one clock cycle so that each processor can access memory and ALUs once each basic minor cycle. Since each processor has its own register set it can run independently at its normal speed. The system thus pipelines four processors in parallel.

Each register of Figure 12 is associated with one of four groups of registers and each group facilitates the multiprocessor operation and is associated with a processor (1-4) of Figure 13. Each of the four groups includes one base pointer register, effective address register, instruction pointer register, top of stack register, stack pointer register, return pointer register, instruction register, CRC register, next register, and a carry flag. Each related group of registers corresponds to one of the four processors. Each processor executes instructions in minor cycles, each minor cycle consisting of four clock cycles. During the first clock cycle a processor will gate the appropriate registers onto the ABUS, BBUS and CBUS. In the next clock cycle the ALUs will be active generating data from their inputs of the

ABUS, BBUS and CBUS. Memory or I/O will be active during the third clock cycle, with the address coming from the ALU 102a and data either being sourced by memory or the ALU 102b. The fourth and final clock cycle will gate the results from memory or the ALU 102b into the appropriate register via the DBUS.

A processor can be viewed as a wave of data propagating through the sequence described above. At each step the intermediate results are clocked into a set of pipeline registers. By using these pipeline registers it is possible to separate the individual steps in the sequence and therefore have four steps executing simultaneously. The four processors can operate without interfering with one another even though they share the ALUs, memory, I/O and many control circuits.

The control of a processor including the pipelining is best understood from Figure 11. For each processor there is a 3 bit counter and an instruction register. These are shown in Figure 11 as counters 137a through 137d, each of which is associated with one of the instruction registers 125a through 125d, respectively. Each of the instruction registers is loaded through the DBUS. As an instruction register is loaded, the instruction is coupled to a PLA 212. This PLA determines from the instruction how many minor cycles are required to execute the instruction and a 3 bit

binary number is then loaded into the counter 113a or 113b or 113c or 113d, associated with the instruction register 125a, or 125b, or 125c or 125d being loaded. For instance, for a CALL instruction loaded into instruction register 125c, the binary number 010 (indicating three minor cycles) is loaded into counter 137c. (Up to 8 minor cycles can be used for a given instruction, however, only up to 6 minor cycles are used for any of the instructions in the currently preferred embodiment.) The count value "000" is used to cause a new instruction to be fetched.

The count (e.g., 3 bits) in a counter and the instruction (e.g., 12 bits) in its associated instruction register from a 15 bit input to the PLA 136. These 15 bit inputs from each of the respective four sets of count registers and four sets of instruction registers are sequentially coupled to the PLA 136 as will be described. The output of the PLA controls the operation of the processors. More specifically: lines 213 control data flow on the ABUS, BBUS and CBUS; lines 214 control the ALU 102; lines 215 control the memory; (and, as will be described later I/O operation of subsections 107, 108, 109 and 220) and lines 216 control data flow on the DBUS. The specific outputs provided by the PLA 136 for a given instruction is best understood from the instructions set, set forth later in this application. The action taken by the

processors to execute each of the instructions is described with the instruction set.

The outputs from the PLA on lines 213 are coupled directly to the devices controlling data flow on the ABUS, BBUS, and CBUS. The signals controlling the ALU are coupled through a one clock phase delay register 217 before being coupled to the ALU via the lines 214. Since all the registers 217 are clocked at the same rate, the register 217 performs delay functions as will be described. Those signals from the PLA 136 used for memory control are coupled through two stages of delay registers 217 before being coupled to the memory, thus the signals on lines 215 are delayed for two clock phases related to the signals on lines 213. The control signals for the DBUS after leaving the PLA 136 are coupled through 3 sets of delay registers 217 before being coupled to the lines 216 and therefore are delayed three clock phases related to those on lines 213. The registers 217 are clocked at a 6mHz rate, thus when the PLA 136 provides output control signals for a given instruction (e.g., contents of instruction register 125a) the control signals during a first clock phase are coupled to lines 213, during a second clock phase, lines 214; during a third clock phase, 215; and during a fourth clock phase to lines 216. During the first clock phase of each

instruction cycle, the contents of the counter 137a and the instruction register 125a are coupled to the PLA 136. During the second clock phase, the contents of the counter 137b and instruction register 125b are coupled to the PLA 136 and so on for the third and fourth clock phases.

Assume now that instructions have been loaded into the instruction registers 125a through 125d and the counters 137a through 137d have been loaded with the corresponding binary counts for the minor cycles needed to perform each of the instructions. For example, assume that register 125a is loaded with a CALL instruction and that 010 has been loaded into counter 137a. During a first instruction minor cycle, 010 and the 12 bit instruction for CALL are coupled to the PLA 136. From this 15 bit input PLA 136 provides at its output all the control signals needed to complete the first minor cycle of the CALL instruction (e.g., four clock phases) for the ABUS, BBUS CBUS, the ALU, the memory and the DBUS. Since the system uses pipelining multiprocessing, the control signals on lines 213 used to carry out the first clock phase of the CALL instruction which is the inputs to the ALUs.(During this first clock phase the other control lines are controlling the ALU, the memory and the DBUS of other processors, for different instructions in the pipelines.)During

phase 2, the count in counter for 137b and the instruction in register 125b are coupled to the PLA 136. During phase 2, the signals on lines 213 now control the ABUS, BBUS and CBUS inputs to the ALUs for the second processor to carry out the instruction contained in register 125b. During this second clock phase, the signals on lines 214 control the first processor and the ALU to perform the functions needed to carry out the second clock phase of the CALL instruction contained in register 125a. (Note a delay equal to one phase was provided by register 217.) Similarly, during the third phase, the signals on lines 213 control the ABUS, BBUS, and CBUS for the third processor to carry out the instruction contained in register 125c; the signals on lines 214 control the ALU to carry out the instruction contained in register 125b, and the signals on lines 215 control the memory to carry out the instructions in register 125a for the first processor. And, finally, during the fourth clock phase, the instruction from register 125d, along with the count in counter 137d are coupled to the PLA 136. The signals on lines 213 control the ABUS, BBUS and CBUS to carry out the instruction contained within register 125d fourth processor; the signals on lines 214 control the ALU to carry out the instruction in register 125c for the third processor; the signals on lines 215 control the memory to carry

out the instruction in register 125b for the second processor; and the signals on lines 216 control the DBUS to carry out the instruction in register 125a for the first processor.

After four cycles of the 16mHz clock the count in register 137a decrements to 001. Each register is decremented on the clock cycle following the use of the contents of the counters contained by the PLA 136. The input to the PLA 136 thus changes even though the instruction within register 125a is the same. This allows the PLA 136 to provide new output signals needed for the second minor cycle of the CALL instruction. These control signals are rippled through the control through the control lines 213, 214, 215 and 216 as described above. When the count in a counter reaches 000, this is interpreted as an instruction fetch for its associated processor.

Therefore, each of the four processors may simultaneously execute an instruction where each of the instructions has a different number of cycles. The control signals reaching the imaginary line 219 for any given clock cycle represent control signals for four different instructions and for four different processors. For example, the control signals associated with the first processor during a first cycle appear on lines 213; during a second cycle on lines 214; during a third cycle on lines 215; and

during a fourth cycle on lines 216. The control signals needed by the second processor follow behind; those needed by the third and fourth processors following behind those used by the second processor.

The pipelining of the signals is illustrated in Figure 13. The multiprocessor operation of the processor 100 of Figure 10 is shown in Figure 13 as four processors, processors 1, 2, 3 and 4. Each one of the groups of registers is associated with one of the processors. The four phases of a single instruction cycle are shown at the top of Figure 13. In Figure 13, registers 101 are used to indicate that the contents from the specific registers called for in an instruction are placed on the ABUS, BBUS and CBUS. The registers are 118, 119 and 120 on the ABUS; 122, 123, 124 and 125 on the BBUS; 120, 122, 129, 130 and 131 on the CBUS.

During a first phase, signals previously stored in the group 1 registers (e.g., two of them) are gated from the registers onto the ABUS, BBUS and CBUS. While this is occurring, signals associated with the group 2 registers are gated from the registers 141, 142, 143 into the ALU 102a and 102b. This is shown in Figure 13 as processor 2 under the first phase column. Simultaneous signals are gated from registers 145a and 145b into the memory for

group 3 registers for processor 3. And, finally, during this first phase, signals associated with the group 4 registers are gated from registers 146 onto the DBUS. During the second phase, signals associated with a group I registers are coupled from the ALU to registers 145. The data associated with group 2 registers are coupled to memory. The data associated with the group 3 registers is coupled from the register 146 onto the DBUS. Those associated with the group 4 registers are gated onto the ABUS, BBUS and CBUS . And, similarly, during the third and fourth phase of each instruction cycle, this pipelining continues as shown in Figure 13, thus effectively providing four processors.

E. PROCESSOR INSTRUCTIONS

In this section each instruction of the processor is set forth, along with the specific registers and memory operations. Lower case letters are used below to indicate the contents of a register. For example, the contents of the instruction register are shown as "ip". The registers and flags are set forth below with their correlation to Figure 12.

FIGURE 12

		IDENTIFICATION
ip	instruction pointer (14 bits) (fixed range of 0000 - 3FFF) (not accessible to ROM based programs)	120
ir	instruction register (12 bits) (not accessible to ROM based programs)	125
bp	base page pointer (14 bits) (fixed range of 8000 -FFFF) (write only)	118
ea	effective address pointer (16 bits) (not accessible to ROM based programs)	119
sp	data stack pointer (8 bits) (positive offset from bp, grows down)	123
rp	return stack pointer (8 bits) (positive offset from bp, grows up)	124

tos	top of data stack (8 bits)	122
next	item below top of data stack (8 bits)	131
crc	used as scratch or in CRC calculations (8 bits)	130
flags	carry flags; (1 bit) processor ID (2 bits)	129

The top element of the return stack is also addressable as a register, even though it is physically located in RAM.

Instruction Table

CALL	1aaa aaaa aaaa	Subroutine call
CALL lib	0000 aaaa aaaa	Library call
BR	0010 1aaa aaaa	Branch
BRZ	0010 00aa aaaa	Branch on TOS==0

BRC	0010 11aa aaaa	Branch on Carry set
CALL interseg	0011 LLLL LLLL	(Subroutine)
	0000 hhhh hhhh	Two word instructions
LIT	0101 1ffh bbbb	Constant op TOS
LDC	0101 111h bbbb	Load Constant
ALU	0101 00ef ffff	Top of Stack and NEXT
RET	0101 0011 1101	Return or Bit set in other instruction
IN,OUT	0100 0wrr rrrr	Read/Write I/O Register
LD,ST bp+a	0100 1waa aaaa	Load, Store
LD,ST (bp+p)+a	011p pwaa aaaa	Load, Store
LDR,STR r	0101 010w rrrr	Load, Store CPU reg

For each instruction, the operation, encoding and timing are set forth below in standard C language notation.

CALL Call Procedure

Operation:

```
*rp++ = lowbyte (ip);
*rp++ = hibyte (ip);
ip = dest;
```

Encoding:

intra-segment:

```
1 a a a  a a a a  a a a a
dest = ip + a + 1; /* displacement a is always negative */
```

inter-segment:

```
0 0 1 1  L L L L  L L L L
0 0 0 0  H H H H  H H H H

dest = H:L;        /* 16 bit absolute address */
```

library:

```
0000 aaaa aaaa
dest = 0x8000+*(0x8001 + a);     /* table lookup call */
```

Timing:

CALL type	#clocks		specific memory operation
intra-seg	3		
		2	*rp++ = lobyte (ip)
		1	*rp++ = hibyte (ip)
		0	ir = *(ip = *dest)
interseg	5		
		4	lobyte (ea) = *ip++
		3	hibyte (ea) = *ip
		2	*rp++ = lobyte (ip)
		1	*rp++ = hibyte (ip)
		0	ir = *(ip = *dest)
library	4		

3 *rp++ = lobyte (ip)

2 *rp++ = hibyte (ip)

1 ip = dest

0 ir = *ip

BR Branch always

Operation:

ip = dest;

Encoding:

0 0 1 0 1 a a a a a a a a

dest = ip + a + 1; /* displacement a is sign extended */

Timing:

#clocks	specific memory operation
1	
	0 ir = *(ip = dest)

BRC Branch on carry

Operation:

```
if ( CF ) ip = dest;
else      ip++;
```

Encoding:

```
0 0 1 0  0 1 a a  a a a a
dest = ip + a + 1; /*  a is sign extended */
```

Timing:

#clocks	specific memory operation
1	
0	ir = *(ip = dest)
	or
0	ir = *(++ip);

BRZ Branch on TOS==0

Operation:

```
if (tos==0, tos=next, next= *(++sp) )  ip = dest;
else ++ip;
```

Encoding:

```
0 0 1 0  0 0 a a  a a a a
```

```
dest = ip + a + 1; /* displacement a is sign extended */
```

Timing:

#clocks	specific memory operation	
2	1	tos = next; next = *(++sp);
	0	ir = *(ip = dest)

or

0 ir = *(++ip)

LDR Move register to TOS

(includes certain indirect, indexed memory reference)

Operation:

*(sp--) = next;

if (reg) { next = tos; tos = reg }

else { next = bp+TOS or next = (bp+2p)+TOS }

Encoding:

0 1 0 1 0 1 0 0 r r r r

reg = r /* see table */

Timing:

	#clocks	specific memory operation
(if (bp+p)+TOS)	5	
		4 lobyte(ea) = *(bp+2p)
		3 hibyte(ea) =*(bp+2p+1)
(if reg, bp+TOS)	3	
		2 *sp-- = next
		if (reg) next = tos;
		1 if (reg) tos = reg
		else next=bp+TOS, ea+TOS

0 ir = *(++ip)

STR Store TOS to register

(includes certain indirect, indexed memory reference)

Operation:

if (reg) { reg = tos; tos = next; }

else { bp+TOS = next or (bp+2p)+TOS = next }

next = *(++sp);

Encoding:

0 1 0 1 0 1 0 1 r r r r

reg = r /* see table */

Timing:

	#clocks	specific memory operation
(if (bp+p)+TOS)	5	
	4	lobyte(ea) = *(bp+2p)
	3	hibyte(ea) =*(bp+2p+1)
if (reg, bp+TOS)	3	
	2	if (reg) reg = tos; else bp+TOS,ea+TOS=next
	1	if (reg) tos = next; next = *(++sp);
	0	ir = *(++ip);

Register assignments

0 0 0 0	Flags	CF x ID1 ID0	
0 0 0 1	CRC low byte	(high byte in TOS)	
0 0 1 0	lowbyte (bp)	/* write */	
	next ("OVER" instruction)	/* read */	
0 0 1 1	highbyte (bp)	/* write */	
	tos ("DUP" instruction)	/* read */	
0 1 0 0	sp		
0 1 0 1	rp		
0 1 1 0	see RPOP, RPUSH		
0 1 1 1	*(bp+TOS)	/* indexed fetch,store */	
1 0 0 0	*(*(bp+0)+TOS)	/* indexed indirect */	
1 0 0 1	*(*(bp+2)+TOS)	/* indexed indirect */	

1 0 1 0 *(*(bp+4)+TOS) /* indexed indirect */

1 0 1 1 *(*(bp+6)+TOS) /* indexed indirect */

RPOP pop return stack

Operation:

```
*(sp--) = next;
next = tos;
tos = *rp--;
```

Encoding:

0 1 0 1 0 1 0 0 1 1 1 0

Timing:

#clocks	specific memory operation

```
2     *sp-- = next
      next = tos;
1     tos = *rp--;;
0     ir = *(++ip)
```

RPUSH push tos onto return stack

Operation:

```
*(++rp)=tos;
tos = next;
next = *(++sp);
```

Encoding:

0 1 0 1 0 1 0 1 1 1 1 0

Timing:

#clocks	specific memory operation

```
2    *(++rp) = tos;
1    tos = next;
     next = *(++sp);
0    ir = *(++ip)
```

5 IN Move I/O register to TOS

Operation:

```
*(sp--) = next;
next = tos;
tos = reg;
```

Encoding:

0 1 0 0 0 0 r r r r r r

Timing:

#clocks	specific memory operation
3	
	2 *sp-- = next

```
                                    next = tos;
                                1   tos = reg ;
                                0   ir = *(++ip);
```

5 **OUT** Store TOS to I/O register

Operation:

```
reg = tos;
tos = next;
next = *(++sp);
```

Encoding:

0 1 0 0 0 1 r r r r r r

Timing:

#clocks	specific memory operation
3	
2	reg = tos;
1	tos = next;

```
                                    next = *(++sp);
                                0   ir = *(++ip);
```

LDC load constant (into TOS)

Operation:

```
*sp-- = next;
next = tos;
tos = constant;
```

Encoding:

0 1 0 1 1 1 1 H b b b b

```
if (H==0)  constant = 0000:bbbb;
else  constant = bbbb:0000
```

Timing:

#clocks	specific memory operation
3	
2	*(sp--) = next; next = tos;

```
                                1     tos = constant;
                                0     ir = *(++ip)
```

LD (bp+a) load from base page

Operation:

```
*sp-- = next
next = tos
tos = *(bp+source);
```

Encoding:

```
0 1 0 0  1 0 a a  a a a a
source = aa aaaa
```

Timing:

```
#clocks          specific memory operation
  3
                 2     *sp-- = next;
                       next = tos;
```

```
                    1    tos = *(bp+source);
                         0    ir = *(++ip)
```

LD (bp+p)+a load indirect

(TOS with byte addressed by pointer at bp+offset then indexed by TOS)

Operation:

```
*sp-- = next;
next = tos
tos = *(*(bp+2p)+offset);
```

Encoding:

0 1 1 p p 0 a a a a a a a a

offset = aa aaaa

Timing:

#clocks	specific memory operation
5	
4	lobyte(ea) = *(bp+2p)
3	hibyte(ea) =*(bp+2p+1)

```
                                    2    *sp-- = next;
                                         next = tos;
                                    1    tos=*(ea+offset)
                                    0    ir = *(++ip)
```

ST (bp+a) store into base page

Operation:

```
*(bp+dest) = tos
tos = next;
next = *(++sp)
```

Encoding:

```
0 1 0 0  1 1 a a  a a a a
dest = aa aaaa
```

Timing:

#clocks	specific memory operation
3	
	2 *(bp+dest) = tos;

```
1       tos = next;
        next = *(++sp);
0       ir = *(++ip)
```

5 ST (bp+p)+a store indirect

(TOS into byte addressed by pointer at bp+2p offset by a)

Operation:

```
*(*(bp+2p)+offset)=tos;
tos = next;
next = *(++sp)
```

Encoding:

```
0 1 1 p  p 1 a a  a a a a
offset = aa aaaa
```

Timing:

#clocks	specific memory operation
5	

```
                                    4    lobyte(ea) = *(bp+2p)
                                    3    hibyte(ea) =*(bp+2p+1)
                                    2    *(ea+off)=tos
                                    1    tos = next;
                                         next = *(++sp)
                                    0    ir = *(++ip)
```

[ALU Group]

Operation:

```
if (r==1) {
      hibyte(ip) = *rp--;
      lobyte(ip) = *rp--;
      }
pipe = tos;              /* internal processor pipeline */
tos = tos op next;
switch (s) {
      case 0: next = next;          /* typical unary op */
      case 1: next = *(++sp);       /* typical binary op */
      }
```

Encoding:

```
0 1 0 1   0 0 r f   f f f f
op = fffff  /* s equal to high order f bit */
s  = (1==unary op),  (0==binary op)
```

Op Table:

code	operation	carry state
00000	tos + next	arith carry
00001	tos + next + carry	arith carry
00010	next - tos	arith borrow
00011	next - tos - carry	arith borrow
00100	tos - next	arith borrow
00101		
00110		
00111		
01000	tos AND next	unchanged

	01001	tos OR next	unchanged
	01010	tos XOR next	unchanged
	01011		
	01100	drop	unchanged 01101
5	swap-drop	unchanged	
	01110		
	01111	CRC step	unchanged
	10000	asl (TOS)	tos7
	10001	asr (TOS)	0
10	10010	rotate left(tos)	tos7
	10011	rotate right (tos)	tos0
	10100	tos	parity(TOS)
	10101		
	10110		
15	10111	3of6 encode	set if not valid

11000	lsl (TOS)
11001	lsr (TOS)
11010	shift left by 4
11011	shift left by 4
11100	swap
11101	tos (NOP)
11110	NOT(TOS)
11111	3of6 decode

Timing:

s	#clocks	specific memory operation	
1	2 (4)		
	(if r==1)	3	hibyte(ip) = *rp--;)
	(if r==1)	2	lobyte(ip) = *rp--;)
		1	tos = alu output
		0	ir = *(++ip)
0	3 (5)		
	(if r==1)	4	hibyte(ip) = *rp--;)
	(if r==1)	3	lobyte(ip) = *rp--;)
		2	tos = alu output